US005974550A

United States Patent [19]

Maliszewski

[11] Patent Number: 5,974,550

[45] Date of Patent: Oct. 26, 1999

[54] **METHOD FOR STRONGLY AUTHENTICATING ANOTHER PROCESS IN A DIFFERENT ADDRESS SPACE**

[75] Inventor: **Richard L. Maliszewski**, Forest Grove, Oreg.

[73] Assignee: **Intel Corporation**, Santa Clara, Calif.

[21] Appl. No.: **08/989,615**

[22] Filed: **Dec. 12, 1997**

[51] **Int. Cl.**[6] **H04L 9/00**; H04K 1/00

[52] **U.S. Cl.** **713/200**; 713/201; 713/202

[58] **Field of Search** 713/200, 201, 713/202; 380/3, 4, 21, 23, 24, 25

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,521,846	6/1985	Scalzi et al.	711/209
4,811,393	3/1989	Hazard	380/21
4,964,163	10/1990	Berry	380/23
5,311,594	5/1994	Penzias	380/23

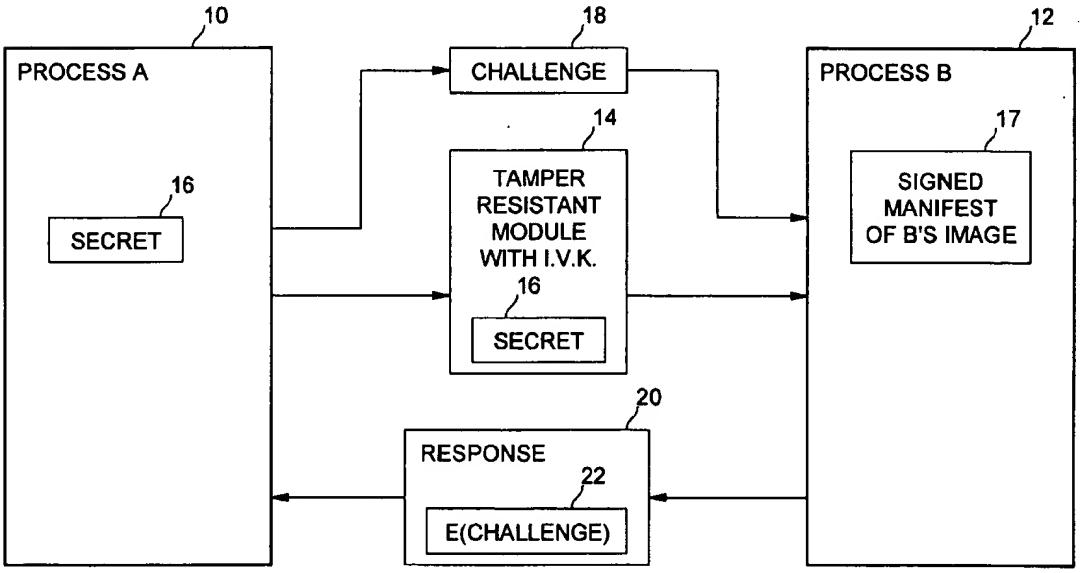
Primary Examiner—Ly V. Hua

Attorney, Agent, or Firm—Steven P. Skabrat

[57] **ABSTRACT**

Authenticating a remote process operating in an address space different than that of a local process includes the steps of creating, by the local process, a tamper resistant module containing a temporary secret, sending the tamper resistant module and a challenge from the local process to the remote process, executing the tamper resistant module by the remote process and recovering the secret when the integrity of the remote process is verified by the tamper resistant module, encoding the challenge using the secret to produce a response, sending the response to the local process, and decoding the response by the local process. Optionally, the tamper resistant module includes a request for information from the second process and the response includes the answer to the request for information.

23 Claims, 3 Drawing Sheets



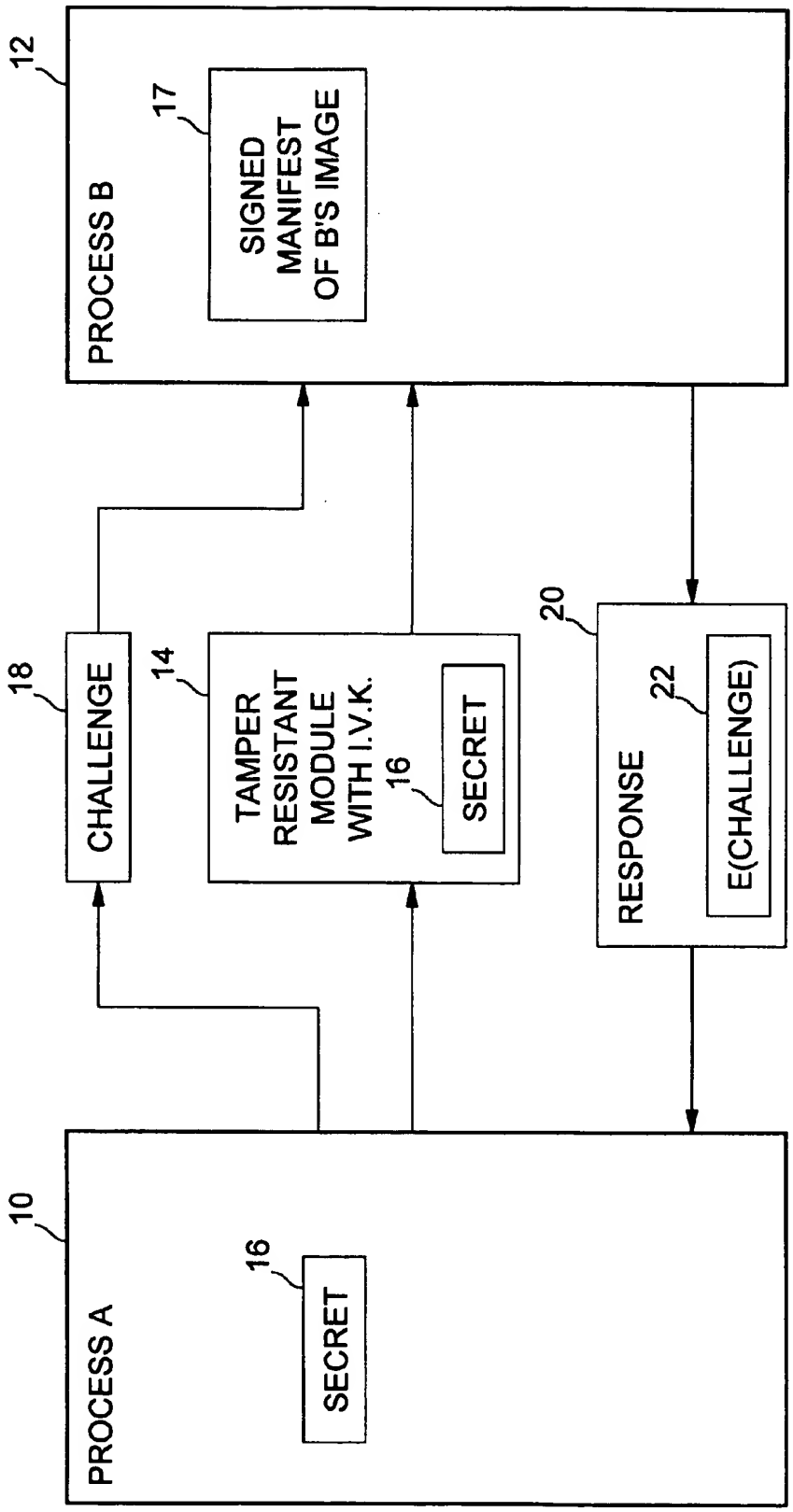

12
PROCESS B
17
SIGNED MANIFEST OF B'S IMAGE
18
CHALLENGE
14
TAMPER RESISTANT MODULE WITH I.V.K.
16
SECRET
20
RESPONSE
22
E(CHALLENGE)
10
PROCESS A
16
SECRET

FIG. 1

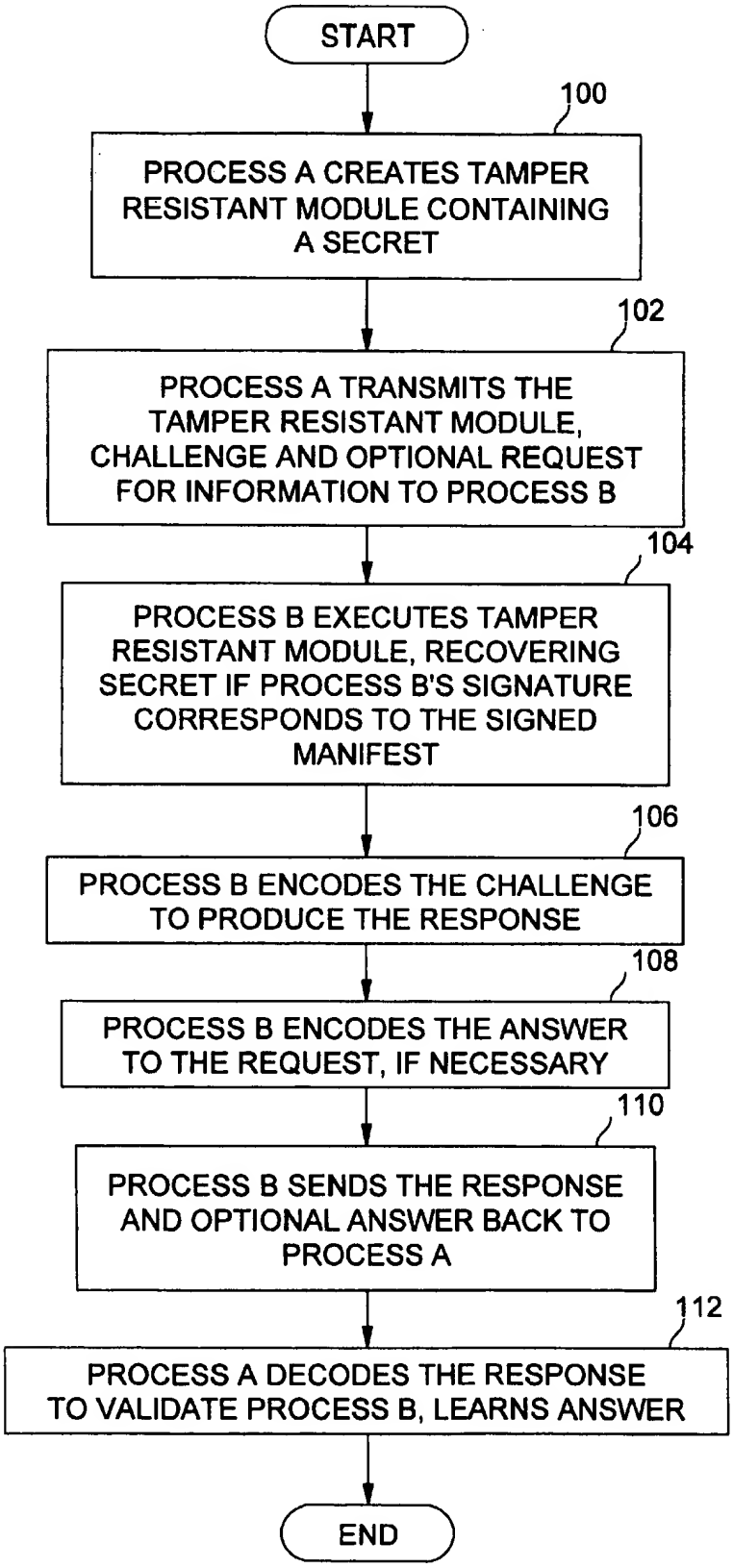

START
100
PROCESS A CREATES TAMPER RESISTANT MODULE CONTAINING A SECRET
102
PROCESS A TRANSMITS THE TAMPER RESISTANT MODULE, CHALLENGE AND OPTIONAL REQUEST FOR INFORMATION TO PROCESS B
104
PROCESS B EXECUTES TAMPER RESISTANT MODULE, RECOVERING SECRET IF PROCESS B'S SIGNATURE CORRESPONDS TO THE SIGNED MANIFEST
106
PROCESS B ENCODES THE CHALLENGE TO PRODUCE THE RESPONSE
108
PROCESS B ENCODES THE ANSWER TO THE REQUEST, IF NECESSARY
110
PROCESS B SENDS THE RESPONSE AND OPTIONAL ANSWER BACK TO PROCESS A
112
PROCESS A DECODES THE RESPONSE TO VALIDATE PROCESS B, LEARNS ANSWER
END

FIG. 2

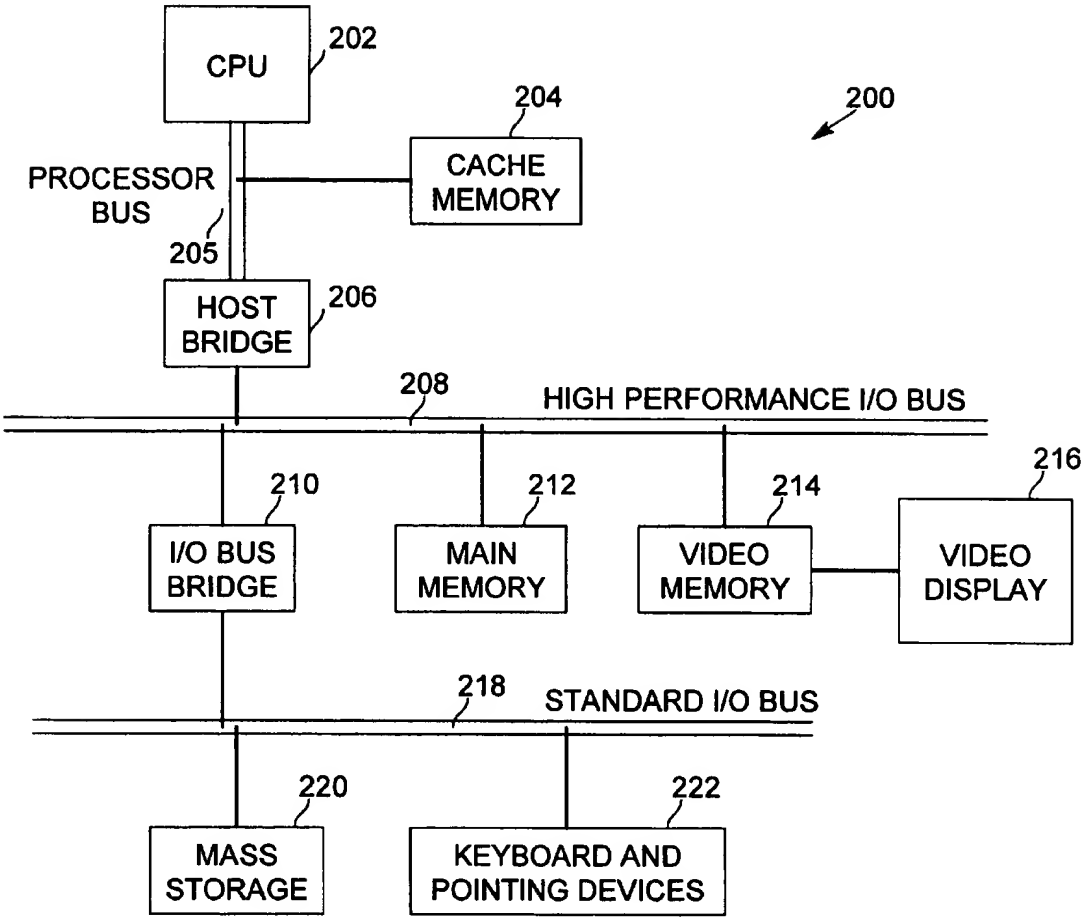

CPU
202
204
200
PROCESSOR BUS
205
CACHE MEMORY
HOST BRIDGE
206
208
HIGH PERFORMANCE I/O BUS
210
212
214
216
I/O BUS BRIDGE
MAIN MEMORY
VIDEO MEMORY
VIDEO DISPLAY
218
STANDARD I/O BUS
220
222
MASS STORAGE
KEYBOARD AND POINTING DEVICES

FIG. 3

METHOD FOR STRONGLY AUTHENTICATING ANOTHER PROCESS IN A DIFFERENT ADDRESS SPACE

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates generally to remote security protocols in computer systems and more specifically to a challenge-response protocol for ensuring the authenticity of a process operating in a different address space.

2. Description of Related Art

Current methods of performing challenge-response protocols in a cryptographic system require the parties or processes performing the protocol to share a high-value persistent secret or both parties must process appropriate pieces of two asymmetric key pairs. In both of these scenarios, prior communication of the secrets between the parties or knowledge of the public component of the other party's key pair is required. Most versions of such challenge-response protocols are vulnerable to "man-in-the-middle" attacks, which are particularly problematic if the two parties are communicating over a network. Hence, general challenge-response protocols prove only that a verified channel between two endpoints sharing a secret has been set up, but one of the endpoints could be insecure. It would be better if one party could authenticate the other party to ensure that the other party has not be tampered with or "hacked", as opposed to just validating that the other party shares the secret. This can be done when the parties share the same address space by checking the contents of memory of the other party, computing its digital signature, and verifying its integrity. However, this cannot be accomplished across different process address spaces unless the memory is shared. Various challenge-response protocols and their deficiencies are described in "Applied Cryptography", by Bruce Schneier, second edition, 1996. What is needed is a protocol that overcomes the above problems and deficiencies of the prior art by securely communicating a required secret as part of an authentication process such that the secret can be a nonce of no persistent worth and wherein the secret need not be previously communicated.

SUMMARY OF THE INVENTION

An embodiment of the present invention is a method for authenticating a first process operating in an address space different than that of a second process. The method includes the steps of creating, by the second process, a module containing a secret, sending the module and a challenge from the second process to the first process, executing the module by the first process and recovering the secret when the integrity of the first process is verified by the module, encoding the challenge using the secret to produce a response, sending the response to the second process, and decoding the response by the second process.

BRIEF DESCRIPTION OF THE DRAWINGS

The features and advantages of the present invention will become apparent from the following detailed description of the present invention in which:

FIG. **1** is a diagram of two processes communicating with each other in a secure manner according to the present invention;

FIG. **2** is a flow diagram of the steps for authenticating another process according to the present invention; and

FIG. **3** illustrates a sample computer system suitable to be programmed with the authentication method in accordance with embodiments of the present invention.

DETAILED DESCRIPTION OF THE PRESENT INVENTION

In the following description, various aspects of the present invention will be described. However, it will be apparent to those skilled in the art that the present invention may be practiced with only some or all aspects of the present invention. For purposes of explanation, specific numbers, materials and configurations are set forth in order to provide a thorough understanding of the present invention. However, it will also be apparent to one skilled in the art that the present invention may be practiced without the specific details. In other instances, well known features are omitted or simplified in order not to obscure the present invention.

FIG. **1** is a diagram of two processes communicating with each other in a secure manner according to the present invention. FIG. **1** shows two processes, Process A **10** and Process B **12**. These processes do not share an address space and may or may not exist on the same processor or computer system. Typically, the processes will be parties to a transaction such as for electronic commerce, wherein Process B **12** asks Process A **10** for a service to be performed. An example of such a service is a consumer purchase, although any transaction desired to be secure involving two parties connected by a network is contemplated. Before Process A fulfills Process B's request for service, Process A seeks to establish the authenticity and verify the integrity of Process B. Authenticity means that the entity purporting to be Process B is actually Process B and not a fraudulent "man in the middle" attacker. Integrity verification means that Process B has not been altered or "hacked" in any way. In an embodiment of the present invention, Process A creates a tamper resistant module **14** that is capable of verifying the integrity of Process B. The tamper resistant module is a small piece of executable software that can be easily communicated from one process to another and executed by a receiving system. Process A creates a temporary secret, passes it in the tamper resistant module to Process B such that the tamper resistant module will only reveal the secret if Process B's credentials and actual image in memory agree. Process B's credentials may be sent along with the tamper resistant module if Process A learns of Process B's credentials beforehand, or they may be examined at Process B's system by the tamper resistant module (i.e., the location of the credentials on process B's system could be passed as an argument to the tamper resistant module before it is executed). The credentials typically will be a predetermined signed manifest of Process B's code image.

Tamper resistant software is software which is resistant to observation and modification. It can be trusted, within certain bounds, to operate as intended even in the presence of a malicious attack. Tamper resistant software must be position independent and not require relocation in memory. Therefore, tamper resistant software does not have to run in the same address space or processor in which it was created. The software is generated by using a tamper resistant compiler (not shown). The tamper resistant compiler is a compiler that, when applied to a well prepared software module, replaces the plain-text source code compiler generated image with a new image that is obfuscated. This self-decrypting software will only execute properly if no part of the image has been altered from the time it was compiled by the tamper resistant compiler. The tamper resistant compiler is a software approach towards providing kernels of software with the ability to run in a "hidden" execution mode. Attempts to decipher what the software is actually doing, or modifications made to the software, will

result in the complete failure of the kernels (i.e., it will not decrypt properly).

The tamper resistant module **14** includes an Integrity Verification Kernel (IVK). An IVK is software that verifies that a program image corresponds to a supplied digital signature. This provides a robust mechanism for detecting changes made to executing software, where those changes might be caused by transmission errors or malicious attacks to the software. Any change to the software results in a failure in the verification process. IVKs for tamper resistant software are constructed to perform self-checks of object code, bilateral authentication of partner modules, and checks on local and remote data to verify the integrity of a software module. The IVK is self-modifying, self-decrypting, and installation unique. Two interprocess software modules requiring to communicate with each other can establish that the module one is calling is indeed the one it is expecting by computing the digital signature of the called module and comparing the computed signature against a predetermined value. This process is called bilateral authentication. In the present invention, the IVK is used to verify the integrity of Process B. Detailed methods for creating the tamper resistant module and providing integrity verification processing with IVKs and bilateral authentication are disclosed in pending U.S. patent applications entitled "Tamper Resistant Methods and Apparatus", Ser. No. 08/662,679, and "Tamper Resistant Methods and Apparatus", Ser. No. 08/924,740, both of which are commonly assigned to the same entity as the present invention and are incorporated herein by reference.

The tamper resistant module **14** also contains a secret **16** which the tamper resistant module will not divulge until the integrity verification for Process B **12** is a success. The secret can be any arbitrary digital value selected by Process A. Process A **10** passes the tamper resistant module **14** containing the secret to Process B **12**, along with a challenge **18**. The tamper resistant module can be sent to Process B before the challenge, after the challenge, or simultaneously with the challenge. Process B then executes the tamper resistant module on its own system. The tamper resistant module is very difficult to examine or change while it is being executed. If the tamper resistant module runs properly and Process B's current image in memory is successfully verified according to the signed manifest of Process B's image **17**, then Process B recovers the secret and uses the secret to encode the provided challenge to produce the appropriate response. Thus, response **20** includes an encrypted challenge E(challenge) **22**. The response is sent back to Process A. Process A, already knowing the secret, decrypts the encrypted challenge. If the response is what Process A expects, then Process A knows that Process B is authentic and its integrity is intact. Further interaction between the parties can then commence.

The present invention is more effective if the secret **16** is a temporary, low value secret that is only used once. It is preferable that the secret is a nonce of no persistent worth. Therefore, Process A changes the secret for every challenge-response sequence. Additionally, a time out mechanism can be employed in Process A such that response **20** must be returned within a specified short time period. This deters a malicious user at Process B from intercepting the tamper resistant module and challenge and attempting to break the inherent security described herein, because the attack cannot be successfully carried out in the allotted time, if ever. Optionally, Process A can also request information from Process B as part of the challenge. The information is then encoded in response **20** back to Process A along with the challenge.

FIG. **2** is a flow diagram of the steps for authenticating another process according to the present invention. At step **100**, Process A creates a tamper resistant module containing a secret. The secret may be any arbitrary digital value known to Process A. Preferably it is a temporary, low-value secret such as a newly generated random number. At step **102**, Process A transmits the tamper resistant module containing the secret, a challenge, and optionally, a request for information to Process B. Next, at step **104**, Process B executes the tamper resistant module, thereby recovering the secret if Process B's digital signature computed by the executing tamper resistant module corresponds to a signed manifest for Process B's image. If the Process B's image failed to verify according to the signed manifest, then the tamper resistant module detects tampering or another error condition, and no further processing is performed by either Process A or Process B. Process A then assumes that Process B is not authentic or that a malicious user has attempted to "hack" the tamper resistant module. Otherwise, Process B encodes the challenge received from Process A to produce the response at step **106**. The encoding operation includes performance of any suitable cryptographic algorithm using the secret as the key. At step **108**, Process B encodes the answer to Process A's request for information, if necessary. Next, Process B sends the response and optional answer back to Process A at step **110**. Finally, at step **112**, Process A decodes the response to validate B and optionally, learns the answer to its information request. If the decoded response contains the challenge Process A sent to Process B, then Process B has been authenticated.

One skilled in the art can see that the present invention securely sends the secret as part of the tamper resistant validation process, and that the secret can be a nonce of no persistent worth. Furthermore, the secret was not required to be shared ahead of time by the communicating processes, and two asymmetric key pair components were not shared by the processes.

In the present invention, the party desiring authentication of a remote process sends an agent (i.e., the tamper resistant module) to the remote process to determine authenticity. This agent is constructed in such a way that it cannot be changed by a malicious user. One limitation, however, of the embodiment shown is that Process A must be able to construct a tamper resistant module that can be executed on Process B's operating system and processor. If the operating systems and underlying processors are the same (for example, WINDOWS NT or WINDOWS 95 operating systems available from Microsoft Corporation, Redmond, WA, and PENTIUM® or PENTIUM® II processors, available from Intel Corporation, Santa Clara, Calif.), then a tamper resistant module created by one party will be a valid executable on the other party's system.

The present invention could also be used in a bilateral manner, wherein Process A authenticates Process B and Process B also authenticates Process A. Both authentications are performed using the above described embodiment. That is, each process sends the other a tamper resistant module and challenge, and the receiving party must send back the correct response to the other party. Thus, each party trusts the other party for further communications.

FIG. **3** illustrates a sample computer system suitable to be programmed with the authentication method in accordance with embodiments of the present invention. Sample computer system **200** may be used to execute processing steps described above for Process A, Process B, or both. When Process A and Process B are on systems remote from each other, Process A is executing on a first sample computer

system and Process B is executing on a second sample computer system connected to the first sample computer system via a network. Sample computer system **200** includes Central Processing Unit (CPU) **202** and cache memory **204** coupled to each other through processor bus **205**. Sample computer system **200** also includes high performance I/O bus **208** and standard I/O bus **218**. Processor bus **205** and high performance I/O bus **208** are bridged by host bridge **206**, whereas high performance I/O bus **208** and standard I/O bus **218** are bridged by I/O bus bridge **210**. Coupled to high performance I/O bus **208** are main memory **212** and video memory **214**. Coupled to video memory **214** is video display **216**. Coupled to standard I/O bus **218** are mass storage **220**, and keyboard and pointing devices **222**.

These elements perform their convention functions well known in the art. In particular, mass storage **220** is used to provide permanent storage for the executable instructions of the authentication and tamper resistant programs/applications, whereas main memory **212** is used to temporarily store the executable instructions of authentication and tamper resistant programs/applications during execution by CPU **202**.

While this invention has been described with reference to illustrative embodiments, this description is not intended to be construed in a limiting sense. Various modifications of the illustrative embodiments, as well as other embodiments of the invention, which are apparent to persons skilled in the art to which the inventions pertains are deemed to lie within the spirit and scope of the invention.

What is claimed is:

1. A method of authenticating a first process operating in an address space different than that of a second process comprising:

creating, by the second process, a tamper resistant module containing a secret;

sending the tamper resistant module and a challenge from the second process to the first process;

executing the tamper resistant module by the first process and recovering the secret when the integrity of the first process is verified by the tamper resistant module;

encoding the challenge using the secret to produce a response;

sending the response to the second process; and

decoding the response by the second process.

2. The method of claim **1**, wherein the tamper resistant module comprises an integrity verification kernel to verify the integrity of the first process.

3. The method of claim **1**, wherein the secret is used only once.

4. The method of claim **1**, wherein the secret is a nonce of no persistent worth.

5. The method of claim **1**, further comprising determining that the first process is not authentic when the response is not received within a predetermined period of time.

6. The method of claim **1**, wherein the first process is verified by the module when a digital signature of the first process determined by the module corresponds to a signed manifest of the first process.

7. The method of claim **1**, wherein the challenge comprises a request for information from the first process.

8. The method of claim **7**, further comprising encoding an answer to the request for information in the response.

9. The method of claim **8**, further comprising decoding the answer by the second process.

10. An apparatus for authenticating a first process operating in an address space different than that of a second process comprising:

a processing unit for executing programming instructions; and

a storage medium having stored therein a plurality of programming instructions of the second process to be executed by the processing unit, wherein when executed, the plurality of programming instructions create a tamper resistant module containing a secret, create a challenge, send the tamper resistant module and the challenge to the first process, receive a response to the challenge from the first process, and decode the response.

11. The apparatus of claim **10**, wherein the tamper resistant module comprises an integrity verification kernel to verify the integrity of the first process.

12. The apparatus of claim **10**, wherein the challenge comprises a request for information from the first process.

13. The apparatus of claim **10**, wherein the secret is a nonce of no persistent worth.

14. An apparatus for authenticating a first process operating in an address space different than that of a second process comprising:

a processing unit for executing programming instructions; and

a storage medium having stored therein a plurality of programming instructions of the first process to be executed by the processing unit, wherein when executed, the plurality of programming instructions receive a tamper resistant module from the second process, initiate execution of the tamper resistant module, recover a secret embedded in the tamper resistant module when the integrity of the first process is verified during execution of the tamper resistant module, receive a challenge from the second process, encode the challenge using the secret to produce a response, and send the response to the second process.

15. The apparatus of claim **14**, wherein the tamper resistant module comprises integrity verification kernel to verify the integrity of the first process.

16. The apparatus of claim **14**, wherein the first process is verified when a digital signature of the first process determined by the module corresponds to a predetermined signed manifest of the first process.

17. The apparatus of claim **14**, wherein the challenge comprises a request for information from the first process and the plurality of programming instructions further comprise encoding an answer to the request for information in the response.

18. An apparatus for authenticating a first process operating in an address space different than that of a second process comprising:

a processing unit for executing programming instructions; and

a storage medium having stored therein a plurality of programming instructions to be executed by the processing unit, wherein when executed, the plurality of programming instructions create a tamper resistant module containing a secret, create a challenge, send the tamper resistant module and the challenge to the first process, initiate execution of the tamper resistant module in the address space of the first process, recover the secret embedded in the tamper resistant module when the integrity of the first process is verified during execution of the tamper resistant module, receive the challenge from the second process, encode the challenge using the secret to produce a response, send the response to the second process, receive the response to the challenge from the first process, and decode the response.

19. An apparatus for bilateral authentication of local and remote processes comprising:

a processing unit for executing programming instructions; and

a storage medium having stored therein a plurality of programming instructions of a local process to be executed by the processing unit, wherein when executed, the plurality of programming instructions create a first tamper resistant module containing a first secret, create a first challenge, send the first tamper resistant module and the first challenge to a remote process, receive a first response to the first challenge from the remote process, decode the first response, receive a second tamper resistant module, initiate execution of the second tamper resistant module, recover a second secret embedded in the second tamper resistant module when the integrity of the local process is verified during execution of the second tamper resistant module, receive a second challenge from the remote process, encode the second challenge using the second secret to produce a second response, and send the second response to the remote process.

20. A machine readable medium having stored therein a plurality of machine readable instructions designed to be executed by a processor, the machine readable instructions for creating a tamper resistant module containing a secret, creating a challenge, sending the tamper resistant module and the challenge to a remote process, receiving a response to the challenge from the remote process, and decoding the response.

21. A machine readable medium having stored therein a plurality of machine readable instructions designed to be executed by a processor, the machine readable instructions for receiving a tamper resistant module from a remote process, initiating execution of the tamper resistant module, recovering a secret embedded in the tamper resistant module when the integrity of a local process is verified during execution of the tamper resistant module, receiving a challenge from the remote process, encoding the challenge using the secret to produce a response, and sending the response to the remote process.

22. A machine readable medium having stored therein a plurality of machine readable instructions designed to be executed by a processor, the machine readable instructions for creating a tamper resistant module containing a secret, creating a challenge, sending the tamper resistant module and the challenge to a first process, initiating execution of the tamper resistant module in the address space of the first process, recovering the secret embedded in the tamper resistant module when the integrity of the first process is verified during execution of the tamper resistant module, receiving the challenge from a second process, encoding the challenge using the secret to produce a response, sending the response to the second process, receiving the response to the challenge from the first process, and decoding the response.

23. A machine readable medium having stored therein a plurality of machine readable instructions designed to be executed by a processor, the machine readable instructions for creating a first tamper resistant module containing a first secret, creating a first challenge, sending the first tamper resistant module and the first challenge to a remote process, receiving a first response to the first challenge from the remote process, decoding the first response, receiving a second tamper resistant module from the remote process, initiating execution of the second tamper resistant module, recovering a second secret embedded in the second tamper resistant module when the integrity of a local process is verified during execution of the second tamper resistant module, receiving a second challenge from the remote process, encoding the second challenge using the second secret to produce a second response, and sending the second response to the remote process.

* * * * *

US005907621A

United States Patent [19]

Bachman et al.

[11] Patent Number: 5,907,621

[45] Date of Patent: May 25, 1999

[54] **SYSTEM AND METHOD FOR SESSION MANAGEMENT**

[75] Inventors: **Margaret A. Bachman**, Charlotte; **Brian D. Jessup**; **Timidthy Sanford McKnight**, both of Huntersville; **Don Cameron Shoff**, Hrrisburg, all of N.C.

[73] Assignee: **International Business Machines Corporation**, Armonk, N.Y.

[21] Appl. No.: **08/749,680**

[22] Filed: **Nov. 15, 1996**

[51] **Int. Cl.**[6] **H04K 1/00**

[52] **U.S. Cl.** **380/25**; 380/49; 395/200.33; 395/200.59; 395/187.01

[58] **Field of Search** 380/25, 49, 24; 395/186–188.01, 200.59, 200.48, 200.33, 200.49; 705/26, 27

[56] **References Cited**

U.S. PATENT DOCUMENTS

5,434,918	7/1995	Kung et al.	380/25
5,542,046	7/1996	Carlson et al.	395/186
5,684,951	11/1997	Goldman et al.	395/188.01
5,708,780	1/1998	Levergood et al.	395/200.12

OTHER PUBLICATIONS

David M. Kristol and Lou Montulli "Proposed HTTP State Management Mechanism", 16 Feb. 1996.

Neil Randall "The New Cookie Monster", PC Magazine 22 Apr. 1997.

Primary Examiner—Gail O. Hayes

Assistant Examiner—Pinchus M. Laufer

Attorney, Agent, or Firm—K. O. Hesse

[57] **ABSTRACT**

Apparatus and method is disclosed for providing user session continuity over several transactions being conducted on the internet. A secure token is made part of each HTML page sent to a user from the server computer and the token is returned to the server with each submitted transaction request. The token is compared with token information originally sent out. The submitted request is recognized as being from an authorized user if the token and a session table have the same information.

15 Claims, 7 Drawing Sheets

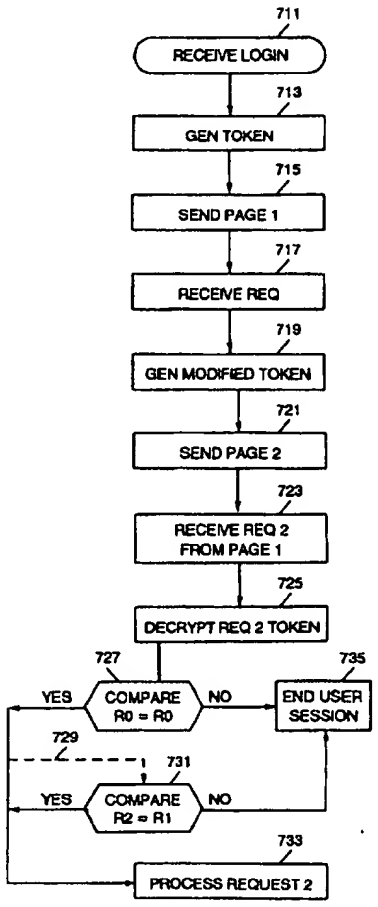


FIG. 1
PRIOR ART

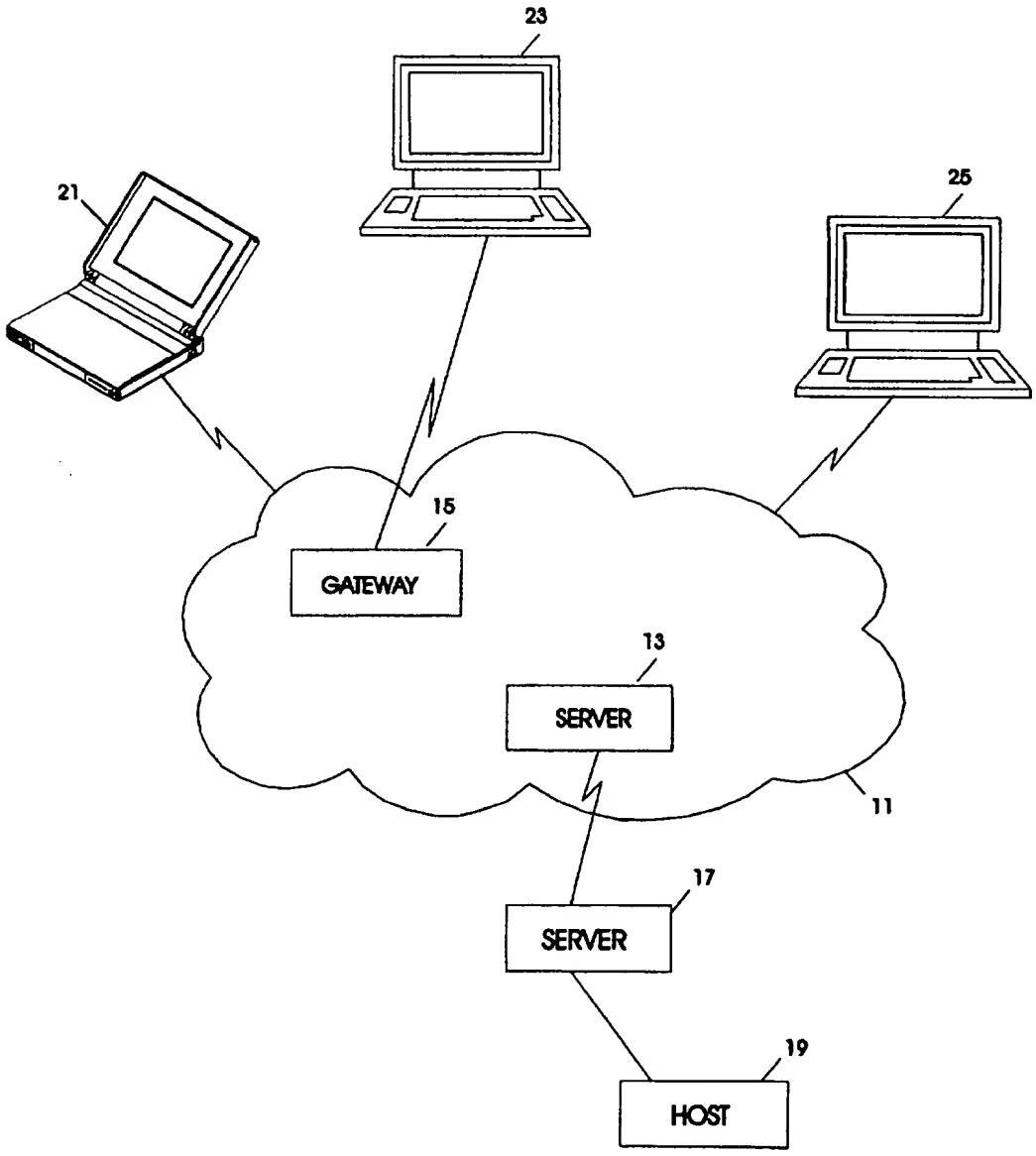
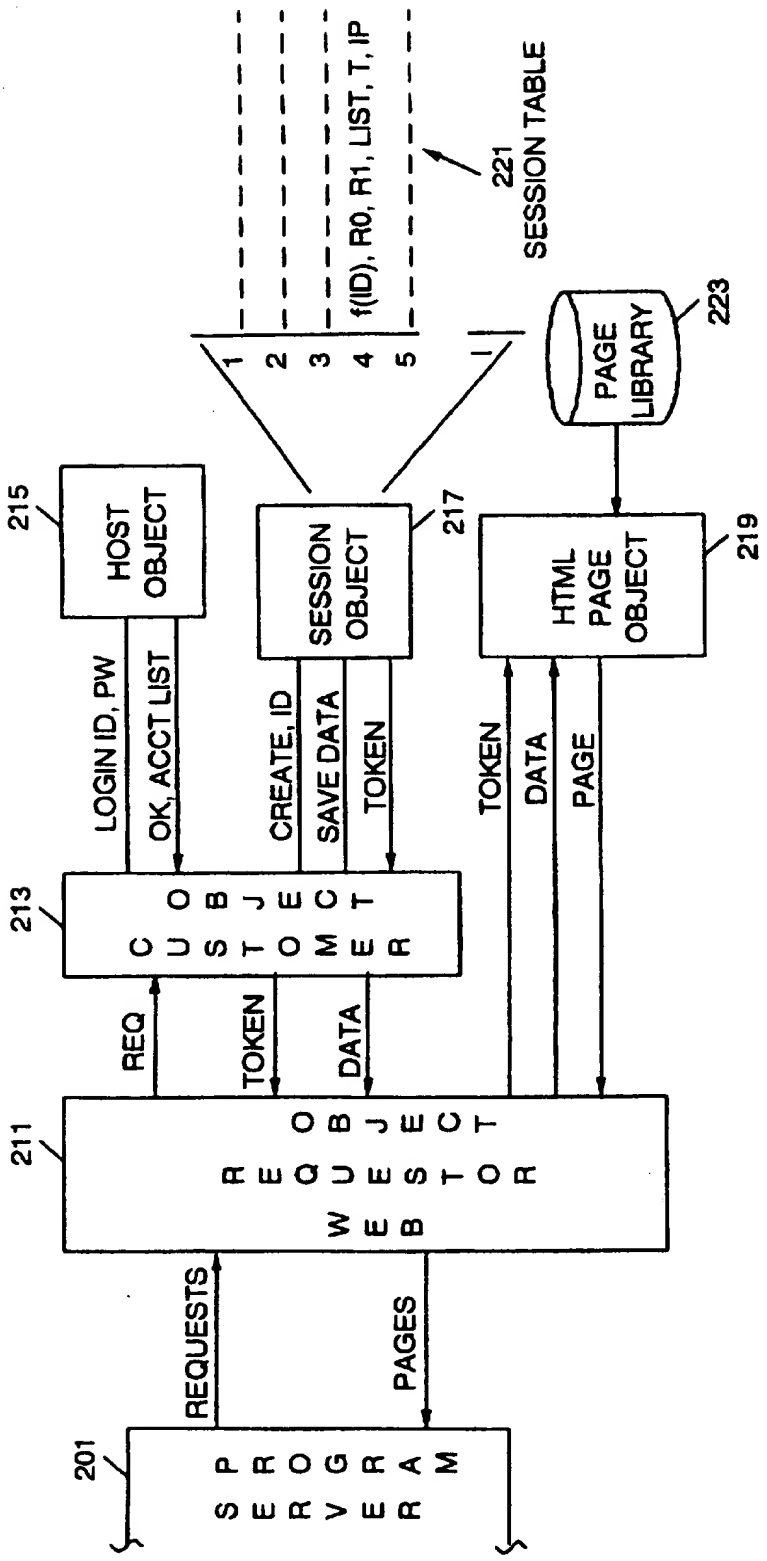


FIG. 2


221
SESSION TABLE
1
2
3
4
5
f(ID), R0, R1, LIST, T, IP
223
PAGE LIBRARY
215
HOST OBJECT
217
SESSION OBJECT
219
HTML PAGE OBJECT
LOGIN ID, PW
OK, ACCT LIST
CREATE, ID
SAVE DATA
TOKEN
TOKEN
DATA
PAGE
213
CUSTOMER OBJECT
REQ
TOKEN
DATA
211
WEB OBJECT REQUESTOR
REQUESTS
PAGES
201
SERVER PROGRAM

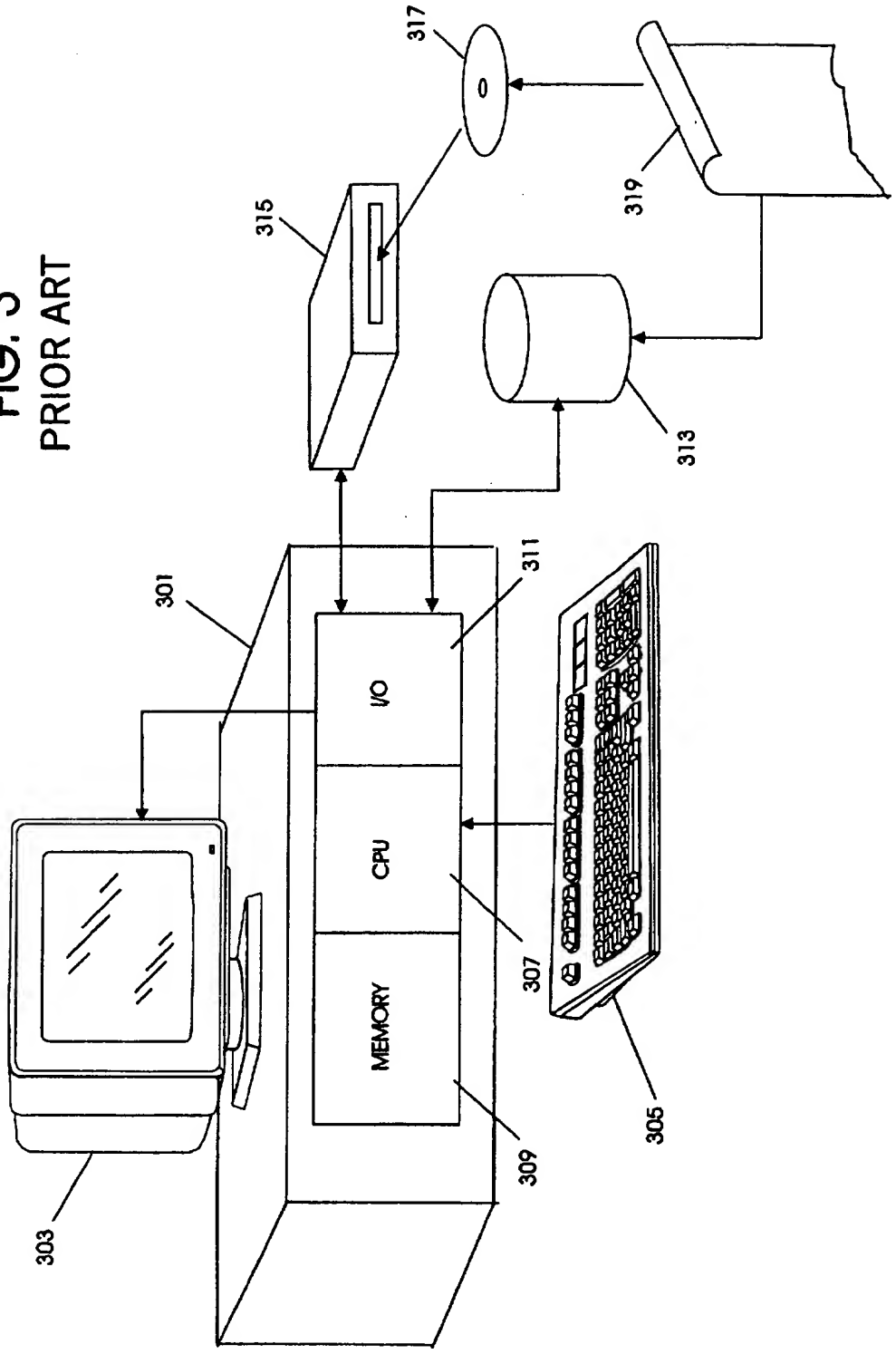


FIG. 3
PRIOR ART

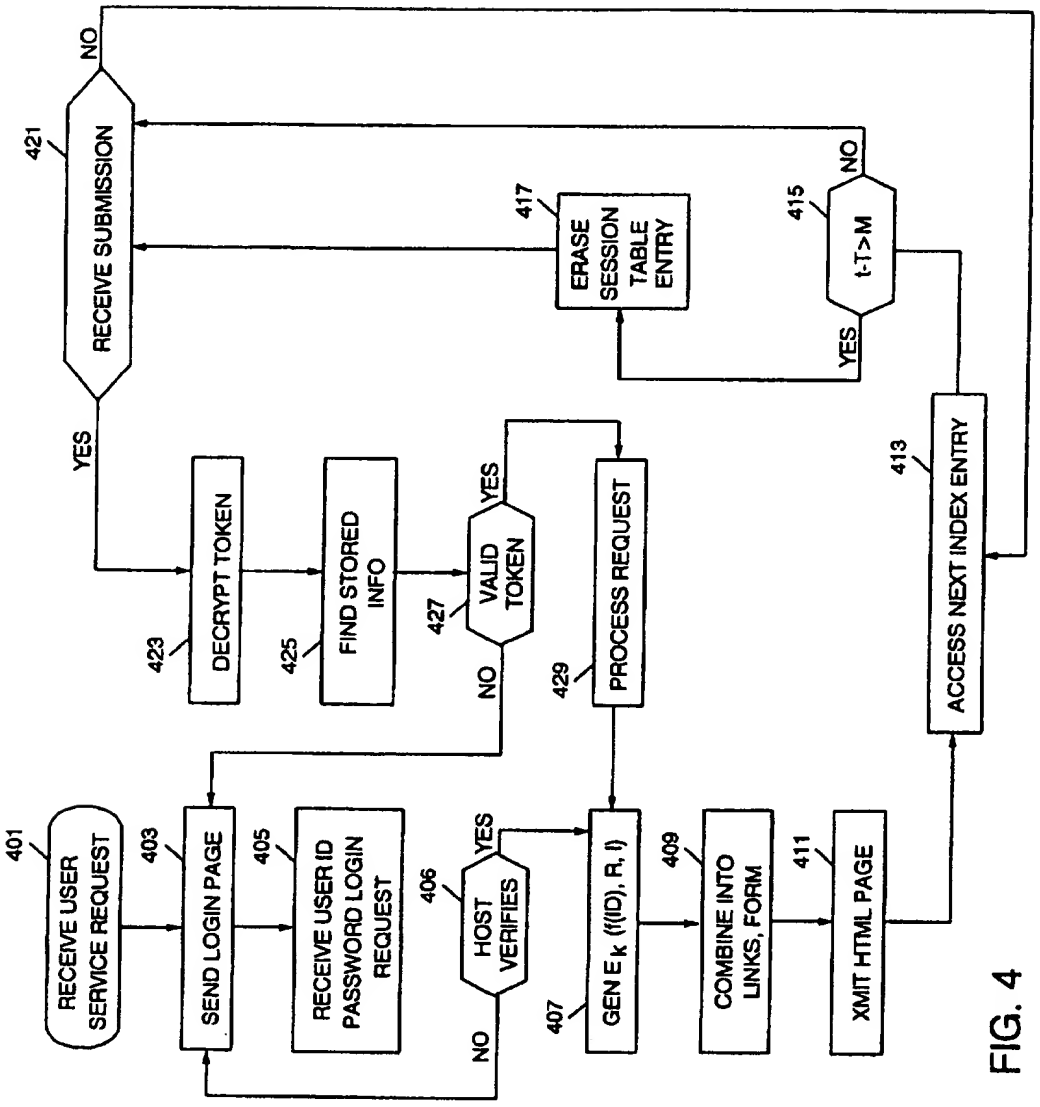

RECEIVE USER SERVICE REQUEST
401
SEND LOGIN PAGE
403
RECEIVE USER ID PASSWORD LOGIN REQUEST
405
HOST VERIFIES
406
YES
NO
GEN E_k (f(ID), R, I)
407
COMBINE INTO LINKS, FORM
409
XMIT HTML PAGE
411
ACCESS NEXT INDEX ENTRY
413
t-T>M
415
NO
YES
ERASE SESSION TABLE ENTRY
417
RECEIVE SUBMISSION
421
NO
YES
DECRYPT TOKEN
423
FIND STORED INFO
425
VALID TOKEN
427
YES
NO
PROCESS REQUEST
429

FIG. 4

FIG. 5

```
<TITLE>
First Internet Bank of CyberSpace - Account Summary
</TITLE>
</HEAD>
<BODY onLoad="timeoutID=setTimeout('timeoutAlert()', 150000)"
background=/icons/back1.jpg>
<CENTER>

<P><B>[                                                     501
  <A
HREF="/cgi-bin/ifsewwwc/ifsewwwc?Logoff&token=0101DDA110050B08&nls=EN">Home<  /A
>  |                                                  503
  <A
HREF="cgi-bin/ifsewwwc/ifsewwwc?Transfer&token=0101DDA110050B08&nls=EN">Trans
fer Funds</A>  |                                      505
  <A
HREF="cgi-bin/ifsewwwc/if sewwwc?Services&token=0101DDA110050B08&nls=EN">Custo
mer Service</A>.. |                                   507
  <A
HREF="/cgi-bin/ifsewwwc/ifsewwwc?Menu&help=AccountSummaryHelp.html&token=0101D
DA110050B08&nls=EN">Help</A>  |                       509
  <A
HREF="/cgi-bin/ifsewwwc/ifsewwwc?General&help=Menu.menu&token=0101DDA110050B08
&nls=EN">General Help</A>  |                          511
  <A
HREF="/cgi-bin/ifsewwwc/ifsewwwc?Logoff&token=0101DDA110050B08&nls=EN">Exit</A
>
 ]</B>
</CENTER>
```

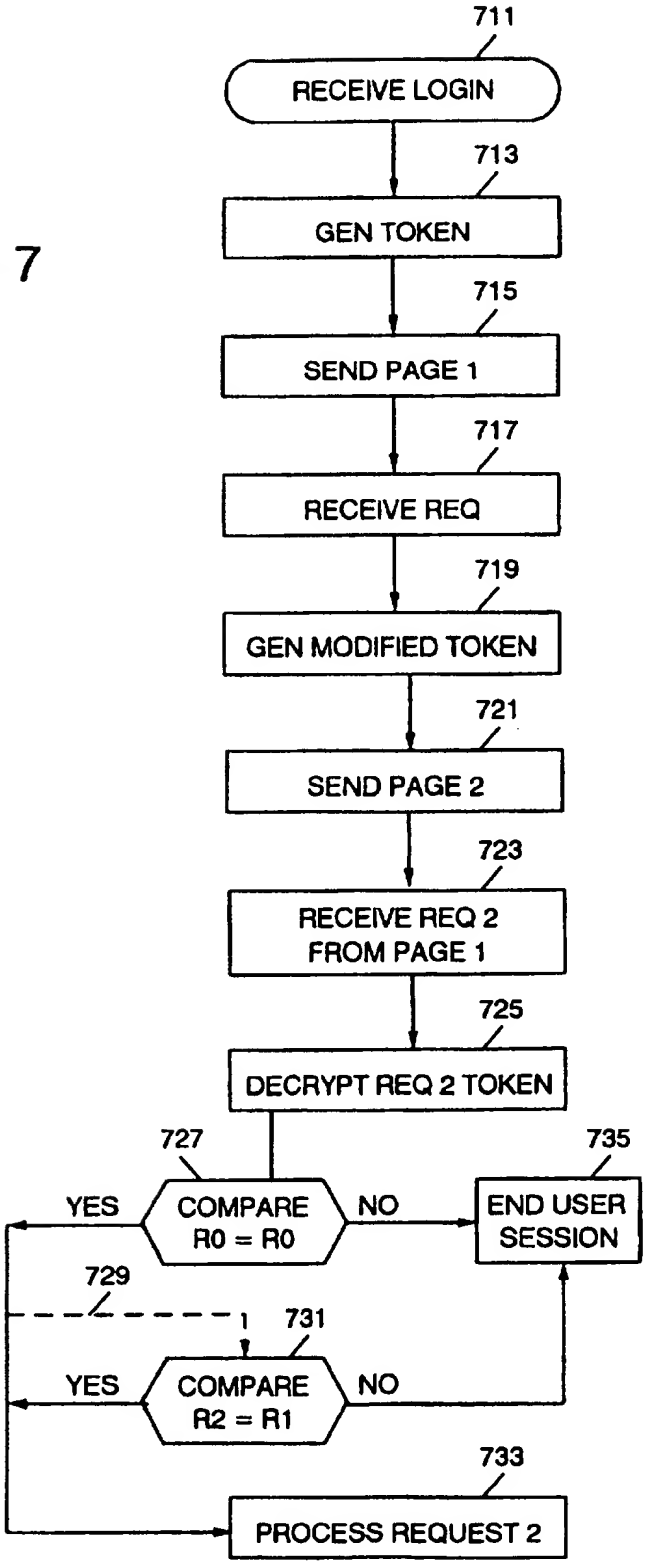
FIG. 6

[Home | Transfer Funds | Customer Service.. | Help | General Help | Exit]

ACCOUNT SUMMARY as of Thu, 17 Oct 1996 13:49:34 EST:

Number	Description	Nickname	Balance
234038	Savings Account	Peter's Savings	$1,764.00
234021	Savings Account		$21,647.00
234081	Checking Account	Marsha's Checking	$13,781.00
234093	Checking Account		$23,801.00
234081	Auto Loan		$17,505.00
234097	Checking Account	Marsha's Checking	$2,286.00
234093	Mortgage	Mike's Home Loan	$3,245.00
234058	Money Manager		$20,246.00
234063	Checking Account		$966.00
234088	Checking Account	Jan's Checking	$18,075.00

FIG. 7


711
RECEIVE LOGIN
713
GEN TOKEN
715
SEND PAGE 1
717
RECEIVE REQ
719
GEN MODIFIED TOKEN
721
SEND PAGE 2
723
RECEIVE REQ 2 FROM PAGE 1
725
DECRYPT REQ 2 TOKEN
727
COMPARE R0 = R0
YES
NO
735
END USER SESSION
729
731
COMPARE R2 = R1
YES
NO
733
PROCESS REQUEST 2

SYSTEM AND METHOD FOR SESSION MANAGEMENT

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to server-oriented programmed apparatus and method for managing a user session on a network where user application connections are not maintained. More particularly, the present invention relates to a token validation apparatus and method that is transparent to the user and client application and is operable with any internet client.

2. Description of Related Art

One technique for conducting important transactions on a public network is to require that a user submit user identity information and a password with every page or form submitted to a server computer at an institution. This may be repetitive and tiresome to a user. Further, the more often that these important values are communicated, the more exposed they become to attack and therefore it is desirable to minimize the number of times that they must be entered and transmitted.

Another known technique for conducting important transactions on a public network includes securing the client or user terminal and encrypting all important messages. A difficulty with securing the client is that many of them are open architecture personal computers which, unlike an automatic teller machine, can not be secured. Further difficulty arises because new browser programs are being made available to the public all the time and a financial institution, for example, can not control the features of these programs beyond the basic standards already being observed. Most of these browser programs store pages in memory and allow scrolling back and forth. Many of these browser programs also allow a page or form received from a financial institution to be stored on disk for future reference. The existence of such pages on an unsecured personal computer, for example at a public library, may allow the next user to recover such information with minimal hacker knowledge and submit the page in a replay mode, pretending to be the authorized user. If the page has been stored on disk, it can be recovered at a later time even if the computer has been turned off.

The use of encryption as described in U.S. Pat. No. 5,416,842 is implemented in many server programs and client browser programs in the form of the secure socket layer (SSL). This method by itself does not provide secure session continuity at an unsecured terminal over the sequential presentation and submission of a number of pages of a multi-page session as provided by the instant invention but only provides protection against those who would attack the communication network.

U.S. Pat. No. 5,237,614 describes a system based upon limiting access by a user to a client at a user location and then relies on encryption to protect the session and provide continuity over the series of pages presented to the user and submitted by the user. This system requires that specialized client software and hardware must be provided at each user location which the instant invention does not require.

The present invention overcomes these inadequacies, problems and disadvantages of the prior art by means of the apparatus and method of the invention which is summarized below.

SUMMARY OF THE INVENTION

An advantage of the present invention is that a secure user session can be established between an internet server and a browser at an unsecured client.

A further advantage of the invention is that no special hardware or software is required at a participating client in order for the method of the invention to perform satisfactorily.

A still further advantage is that the identity of a user is verified in each request submitted by the user, without the need for the user to resubmit identity information and a memorized password with each transaction request.

These and other advantages are obtained by the instant invention through the use of a unique token that is secure from counterfeit and replay attacks. The token is created by combining user identity information with random information and table address information in a way that has a high attack work factor so that it can effectively not be duplicated. To avoid distracting the user, the token is carried in a field of the page that is normally not displayed in the presentation space. Requests submitted by the user automatically include the token which is then compared at the server with the token previously sent out to the user. No steps of the method occur at the client or need be taken by the user making the method transparent to the user. The method and apparatus of the invention operate to complement the transmission security provided by the Secure Socket Layer (SSL) encryption on the internet with a secure user session.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a high level diagram of the internet.

FIG. 2 is a block diagram showing the parts of the invention.

FIG. 3 is a block diagram of a computer which may be used in carrying out the method and implementing the apparatus of the invention.

FIG. 4 is a flow diagram showing the method of the invention.

FIG. 5 shows a portion of a form page wherein the token has been exposed to view.

FIG. 6 shows a portion of a form page as it is displayed to the authorized user for transaction selection.

FIG. 7 shows a flow diagram of an alternate session method according to the invention.

DESCRIPTION OF THE PREFERRED EMBODIMENT

FIG. 1 portrays the overall environment wherein the invention finds utility. The cloud-like communication medium 11 is made up of communication lines and switches connecting servers like server 13 to gateways like gateway 15. Server 13 and gateway 15 provide the communication access to the world wide web internet. Server 13 may be connected to an application server 17 as shown or may be connected directly to a host computer 19 which is the record keeping computer of a financial institution providing transactions over the internet 11 with users located remotely at any of client Personal Computers 21, 23 or 25. Computer 21 may be a lap top computer connected temporarily to communication lines. Computer 23 may be a computer at a public library that is available to members of the library for use in preparing employment resumes and for accessing services and information via the internet 11. Computer 25 may be in a home environment and may be periodically connected via dial up to the internet.

End to end message security is provided for the internet 11 by a secure socket layer (SSL) encryption and decryption facility in each server program in 13 and each browser program in clients 21, 23 and 25. Thus there is little risk of

attack by tapping the phone lines for example. There remains the risk of attack from the client itself after the SSL has provided decryption services. As will be described in greater detail with respect to FIG. **3** wherein a sample computer is shown, each HTML page is stored in the memory of a client to allow forward and backward scrolling. A page can optionally be stored to disk if the user desires. Accordingly pages containing tokens may be available at the client for access by hackers who wish to cause mischief.

FIG. **2** is a block diagram showing a Common Gateway Interface (CGI) program having a number of routines in the form of programmed objects which make up a preferred embodiment of the invention to thwart the above described risks. Web Requestor object **211** runs in server **17** and receives requests and data from a server program **201** such as for example the National Center for Supercomputing Activities (NCSA) server program.

The object **211** also sends pages of data to the server program for communication to the client **23**, for example. An example request and data may be a login with user identification and password from a user at client **23** who wishes to transfer funds from a savings account to a demand deposit account in order to cover purchases about to be made with a cash card.

Object **211** receives the login request and passes it to the customer object **213** which passes it to the host object **215**. The host object **215** passes the user identification and password to host **19** and receives a OK/NOT OK depending upon the correspondence of the password to the user identity information. If the password and identity information correspond, the user is an authorized user and the host **19** also returns an account list and other practical information that can be used by the HTML page object **219** to format a menu page or form to be sent to the authorized user. The user identity information is passed to the session object **217** which sets up a session by generating a session token from a hash of the identity information, from an index and from random numbers R0 and R1. R0 is the primary random number for this session and will not change during the existence of the session. R1 is a transaction random number and it changes every time a new token is generated so that tokens are modified during this session in order to provide randomness and prevent token replay. The identity information, random information R0, R1 and other information such as an account list and the page transmission time T are stored in a session table **221** at the index entry I used to create the token. In an alternate embodiment, to be explained later in greater detail, an Internet Protocol (IP) address is also stored in the session table index entry.

The token is then returned to the customer object **213** and the web requestor object **211** which sends the token and pertinent account data to the HTML page object **219** for layout of a page. The page object **219** retrieves a page template from the page library **223** and inserts the pertinent account information preparatory to sending the page to the authorized user at client **23**. The token is placed into a field that is normally not displayed to a user to avoid the distraction that the token would create in the mind of the user. The token is also placed in each hypertext link in the page. Thus whenever a request is submitted, it will include a token. It is recognized that in an open architecture client, all information is accessible with some knowledge and effort, therefore the security of the token does not rely on being hidden from the user. Instead, it is encrypted. Thus there is, for practical purposes, no way that the user or another person with access to the client can tamper with the token undetected. FIG. **5** shows a portion of HTML source for a menu page with a number of hypertext links **501** through **511** for submission back to the server **17**. Each submission format will include the token.

The web requestor object **211** then passes the formatted page to the server program **201** for transmission to the gateway **15** and client **23** where it is displayed to the authorized user. The authorized user, at the client, peruses the page and provides input by selecting a link item or entering information and returns at least a portion of the page to server **17**.

While the user is perusing the page, the session object periodically scans the content of the index entries comparing the current time with the time T stored in each index entry. If the difference between the current time t and the stored time T exceeds a predetermined amount M, such as for example four minutes, the data at the index entry is erased on the ground that the session has timed out.

At server **17**, a newly submitted request is processed by extracting the token, and decrypting it in the session object **217**. The session uses the index I content of the token to access data stored in the session table **221** at index entry **4**, for example, in FIG. **2** for comparison with the remaining token content. If the comparison indicates that the stored data and the remaining content are equal, the submission is recognized as being from an authorized user. The other portions of the submission are then processed and request information is forwarded to host **19** for action.

While host processing is in progress, the session object changes the transaction random number R1 and generates a new token in preparation for sending transaction result information to the authorized user.

In the event that the comparison indicates that the stored data and the remaining content are unequal or the stored data has been erased, the submission is recognized as being from an unauthorized user or a user whose session has expired. The other portions of the submission are then processed and with the page library **223**, used to create a login request page to be sent to the user.

In another embodiment of the invention, the IP address of the client used to send the login request is made part the session table entry. Thereafter this stored IP address is compared with the IP address from which any submission is received. If the IP address from which a submission was received is different from the IP address in the table, even if the token is otherwise found to be valid, the submission is not considered to be part of a valid session. This further embodiment provides another level of session security without adding significant processing overhead.

FIG. **3** shows an example computer that may be used in clients **21**, **23** and **25** as well as at gateway **15** and servers **13** and **17**. Personal computer clients typically require less extensive resources and less powerful processors than are required by gateway and server computers, but in other respects they are very similar.

The computer comprises the main case **301**, a display **303** and a keyboard **305**. Inside of case **301** is found, random access and read only memory **309**, a central processing unit **307** and input output (I/O) circuits **311**. The I/O circuits are controlled by programs to in turn control the keyboard **305**, display **303** and mass storage in the form of a fixed disk media **313** and a removable disk unit **315**. The unit **315** may be a magnetic diskette unit or a compact optical disk unit. Disk media **317** is removably insertable into unit **315**. Various browser programs and or server programs including the program **319** in which the instant invention is embodied may be written onto disk media **313** and/or diskette media

317 from which it is loaded into memory 309 to control the computer in accordance with the teachings and method of the invention. It will be recognized that media is not limited to storage media but may also be a communication media that carries the program of the instant invention to a server computer like the computer of FIG. 3.

At a client personal computer, each page received from a server is stored in memory 309 of a client to allow forward and backward scrolling. A page can optionally be stored to disk 313 if the user desires. Accordingly pages containing tokens in accordance with the instant invention will be available at the client for access by those who wish to cause mischief. It is for this reason that it is important that there be no need to perform any operations such as decryption or interpretation upon the token at the browser or client. All of the unique apparatus if the invention is located at a server and can interact with any browser observing current Hypertext Transfer Protocol (HTTP) standards.

OPERATION OF THE INVENTION

Referring now to the flow diagram in FIG. 4, the operation of the invention at a server 17 will be described. At block 401, the server 17 receives a request for service from an internet user. In response to this request, the server sends, at block 403, a login page to the user. The user enters identity information such as a user ID and a memorized password that can be used to verify that the person who remembered the password is an authorized user. The user then submits the login page at block 405. At block 406, the host 19 has verified correspondence between the ID and password and at block 407, the session object 217 generates a token as described earlier with respect to FIG. 2.

The token can be represented by the expression:

$$E_k(f(ID),R,I)$$

where E subscript k is the encryption using the encryption key k of the argument value in the parentheses. f(ID) is a function such as a hash function of the users identity information. R is a random number and I is the index to an entry in the session table 221. It will be recognized that the items in the parentheses of the above expression are not crucial to the invention but that other values serving equivalent functions can be substituted therein. For example the token must exhibit randomness to avoid predictability and therefore any value that exhibits a form of randomness may be substituted for the value R. In one embodiment, a session random number R0 and a transaction random number R1 are placed in the parentheses. R1 is changed for each page but R0 remains the same until the session ends. By comparing only R0 to the corresponding R0 stored in the session table at index I, earlier pages can be used by a user to send in requests. If it is desired to limit requests to those from the latest page, both R0 and R1 are compared with corresponding stored values to limit an incoming submitted token to the latest token. The token is generated by first combining the values of the variables in the parentheses into a multi-byte field which is then encrypted using an algorithm such as the DES algorithm. In addition, the values of the variables used to generate the token and the then current time T, are stored in the session table at the index location. In one alternate embodiment, an internet protocol (IP) address is also included in the session table entry. The inclusion of the IP address allows a valid token coming from a different address to be recognized as possibly being used by an un-authorized user.

The token is combined at block 409 into an HTML page by being placed into a hidden or other normally non-displayed field of a page by the HTML page object shown in FIG. 2.

At block 411 the formatted HTML page is transmitted over the internet to the user where it is displayed, soliciting input of transaction selections such as Transfer Funds, Customer Service or other transaction. The token is not displayed in normal operation but exists in each hypertext link on the page. For example, as shown in FIG. 5, "token=0101DDA110050B08" appears in hypertext links 501, 503, 505, 507, 509 and 511 but the tokens do not appear in the display of these options in FIG. 6.

While awaiting the submission of requests in response to outstanding pages, the session object scans the entries in the session table as in block 413 and compares the time T stored in each entry with the now current time t at block 415. If the current time t is much later by a predetermined amount M than the stored time T, the session is considered to have timed out and the data in the entry is erased as depicted in block 417. Such removal of expired sessions from the session table continues until the next submission is received at block 421.

When the authorized user makes a selection, the request is submitted to the server at block 421, retaining therein the token originally sent out in the HTML page. At the server, the server program passes the submission through to the session object where the token is decrypted at block 423 to expose a multi-byte field of token information including an index. The index is used to access the stored token information at block 425. The stored token information is compared at block 427 with the received token information and the request is honored only if the stored and received token information are equal to each other indicating that the received token is a valid unexpired token. Although the index portion of the token information is not itself stored in the session table, the index is used to access the session table and therefore, if the index is not the same as was sent in an outgoing token, an incorrect session table entry will be accessed and it will not compare equal to the remainder of the decrypted token.

If the token is found to be valid at block 427, the transaction is forwarded to the host object for processing at block 429 and a new token is generated at block 407 to be sent to the authorized user with the response page at block 409.

If the token is found to be invalid, or in the alternate embodiment of block 431, if the token was received from an IP address other than stored in the session table, the request is not processed but a login menu is sent to the user at block 403 so as to determine if the user is an authorized user as was previously described with respect to these blocks 403 and 405.

During a session, multiple pages will be in existence at a client. In some instances, a user may make a selection from a much earlier page. Each page contains a token that is generated from the same session table entry but each token is further randomized by including another random number R1 in the value encrypted. In the preferred embodiment, R1 is not made part of the comparison to validate the token. Therefore a user need not submit requests only from the latest page.

This operation is shown more clearly in FIG. 7 where a session login is received at block 711 and a token is generated at block 713 using an R0 and an R1. It is sent in a page one at block 715 and received in a request at block 717. Assuming the returned token is the same as the generated token, a modified token is generated at block 719 using R0 and a new transaction random number R2. It is sent in page two at block 721. A request two is received but it may have been a link from page one. The request two token

is decrypted and the session random number R0 is compared at block 727 to the R0 stored in the session table as describer earlier. If it is desired to limit requests to those linked from the latest page, that is page two, then path 729 is followed and the compare of R1 with R2 is made at block 731. The session is ended at block 735 because an old page was used. In the alternative, if the user is allowed to use earlier pages, block 731 is bypassed and request two is processed at block 733 because the session random number R0 was the same as stored in the session table at index I.

Of course, many additional modifications and adaptations to the present invention could be made in both embodiment and application without departing from the spirit of this invention. For example, although an object oriented programmed server computer is the current preferred embodiment, a fixed embodiment or an alternate programming architecture may be used. Likewise other encrypting algorithms may be chosen to protect the token from reverse engineering at an unsecured client. Accordingly, this description should be considered as merely illustrative of the principles of the present invention and not in limitation thereof.

We claim:

1. Method of managing a user session on a network comprising the steps:

A) receiving at a server, user-unique information from a user-at-a-client;

B) generating at said server, from said user-unique information, a session token;

C) storing said session token at said server;

D) combining said session token into a first page;

E) transmitting said first page to said client for display to said user and for input from said user;

F) receiving a first submitted request at said server;

G) comparing a submitted session token from said first submitted request with said session token at said server to identify said first submitted request as being from said user;

H) processing said submitted request and preparing a second page;

I) generating from said user unique information, a modified token;

J) storing said modified token at said server;

K) combining said modified token into said second page;

L) transmitting said second page to said client for display to said user and for input from said user;

M) receiving a second submitted request at said server;

N) comparing a second submitted token from said second submitted request with said modified token at said server to identify said second submitted request as being from said user in a same session as said first page and said first submitted request.

2. The method of claim 1 wherein said step B) further comprises the steps:

assembling an argument including at least part of said user-unique information and session random information;

encrypting said argument to create said session token;

and wherein said step I) further comprises the steps:

assembling another argument including at least part of said user-unique information, said session random information and second random information;

encrypting said another argument to create said modified token.

3. The method of claim 2 wherein said comparing step N) further comprises the steps:

retrieving said second submitted session token from said second submitted request;

decrypting said second submitted token to retrieve a submitted argument;

comparing said submitted argument with said another argument;

determining that said second submitted request is from said user if session random information in said submitted argument is equal to session random information in said another argument at said server.

4. The method of claim 3 wherein said arguments comprise:

said user-unique information, said session random information and index information, said index information being an index into a session table where said user-unique information and said session random information is stored at said server, said tokens being generated by encrypting a concatenation of said at least part of said user-unique information, said session random information and said index information.

5. The method of claim 2 wherein each of said arguments comprise:

a hash of said user-unique information, said session random information and index information, said index information being an index into a session table where said user-unique information and session random information is stored at said server.

6. The method of claim 5 wherein each of said tokens is generated from a same session table entry and a time of transmission is stored with each entry in said session table.

7. The method of claim 6 further comprising the steps:

processing a submitted form and preparing a third page;

changing said second random information in said session table entry;

generating from said session table entry, a third token;

combining said third token into said third page;

transmitting said third page to said client for display to said user and for input from said user;

receiving a submitted form at said server node;

comparing a submitted session token from said submitted form with said session token at said server node to identify said submitted form as being one of said first page, said second page and said third page from said user in the same session as said first page, said second page and said third page.

8. The method of claim 6 further comprising the steps:

comparing a current time with a time stored with said session table entry at said server;

erasing said stored session table entry when a difference between said current time and said time stored with said session table entry exceeds a predetermined time.

9. The method of claim 8 further comprising the steps:

sending, upon receipt from said user of a request which is from a page for which a session table entry is not found, a page soliciting resubmission of said user-unique information;

generating from said user-unique information, another session token;

combining said another session token into said page for which a session table entry was not found;

transmitting said page for which a session table entry was not found to said client for display to said user and for input from said user.

10. A computer program product for use in a data processing system, the computer program product comprising:

a computer useable medium having computer readable program code embodied therein for managing a user session, said computer program product including:

generating means for responding at a server to user-unique information from a user at a client and generating from said user-unique information, a session token;

means for combining said session token into portions of a first page;

means for transmitting said first page to said client for display to said user and for input from said user;

means for receiving a first submitted request at said server;

means for comparing a submitted session token from said first submitted request with said session token at said server to identify said first submitted request as being from said user;

means for processing said first submitted request and preparing a second page;

means for generating a modified token from said session token by including a transaction random number in said token;

means for combining said modified token into said second page;

means for transmitting said second page to said client for display to said user and for input from said user;

means for receiving a second submitted request at said server;

means for comparing a submitted session token from said second submitted request with said modified token at said server to identify said second submitted request as being from said user in a same session as said first page and said first submitted request.

11. The computer program product of claim 10 wherein said means for comparing further comprises:

means for retrieving a submitted session token from a submitted request;

means for decrypting said submitted token to retrieve a submitted argument;

means for comparing said submitted argument;

means for determining that said submitted request is from said user in a current session if a submitted session random number in said submitted argument is equal to a session random number in an argument from which tokens are being generated for said current session at said server.

12. The computer program product of claim 10 further comprising:

means for processing a submitted request and preparing a subsequent page;

means for generating a modified token from said argument by including a transaction random number in said token;

means for combining said modified token into a subsequent page;

means for transmitting said second page to said client for display to said user and for input from said user;

means for receiving a subsequent submitted request at said server;

means for decrypting a submitted token from said second submitted request to retrieve a submitted argument;

means for comparing said second submitted argument with said argument at said server to identify said submitted request as being from said user in a same session as said page and an earlier submitted request.

13. A server data processing system for managing a session with a client data processing system comprising:

means for generating a taken from token arguments, said token arguments including user-unique information, a session random number and a page random number, said page random number being different for each page causing said token to be different for each page;

means for transmitting said token to said client data processing system as part of each page;

means for storing at said server data processing system, said token arguments and a time of transmission of a page;

means for comparing a session random number, from a submitted token received at said server data processing system in a submitted request from a client data processing system, with a session random number stored at said server data processing system, said submitted token being a token from any of said each page;

means for determining that said submitted request is part of an existing session when said submitted session random number compares equal to said session random number stored at said server data processing system.

14. The server of claim 13 further comprising:

means for periodically testing said time stored with said token arguments and removing said stored token arguments from storage at said server data processing system when a difference between a current time and said time stored with said stored token arguments exceeds a predetermined value.

15. The server of claim 14 further comprising:

session table means for storing said token arguments, said token arguments further including index information;

said means for comparing further comprising means for using index information, from a submitted token received at said server data processing system in a submitted request from a client data processing system, to address an entry in said session table means and retrieve stored token arguments when said stored token arguments have not been removed by said means for periodically testing.

* * * * *